A Roadmap-based Algorithm for Planning Object Handling in Changing Industrial Plants

Moez Cherif, Marc Vidal, Christian Laugier

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A Roadmap-based Algorithm for Planning Object Handling in Changing Industrial Plants

Moëz Cherif, Marc Vidal, Christian Laugier

N ° 3629

February 1999

THÈME 3

ISSN 0249-6399

INRIA
RHÔNE-ALPES

A Roadmap-based Algorithm for Planning Object Handling in Changing Industrial Plants

Moëz Cherif*, Marc Vidal, Christian Laugier

Thème 3 — Interaction homme-machine,
images, données, connaissances

Projet Sharp

Rapport de recherche n ° 3629 — February 1999 — 23 pages

Abstract: This paper addresses the problem of planning collision-free motions for a fleet of mobile robots moving in a cluttered and changing workspace. We focus on the problem of planning handling operations in an industrial plant: *find a collision-free trajectory for a set of mobile robots for handling a group of movable objects in a cluttered industrial plant.* The motion planner we describe copes with constraints due to changing of the workspace by decomposing the planning problem into 2 stages: (1) building a single roadmap by pre-processing the C-spaces of the moving systems, and (2) planning coordinated collision-free paths for these systems including manipulation of the roadmap to guarantee the incorporation of the workspace changing. The planner has been implemented and demonstrated on 2D models of industrial plants.

Key-words: Motion planning, object handling, multi-robot systems, mobile robots, obstacle avoidance, changing environment.

(Résumé : tsvp)

This work was motivated by the problem of automating intervention and maintenance tasks in nuclear power plants. Problem communicated by Électricité de France (EDF).

The author gratefully acknowledges Dr Emmanuel Mazer of GRAVIR-CNRS Lab., Grenoble, for his comments on this work. Discussions with members of the Research & Study Division of EDF (DER-EDF) were also appreciated.

This report has been submitted to the International Journal of Robotics and Automation (IASTED).

* Visiting research scientist, on leave from Simon Fraser University, Burnaby, Canada. e-mail: `Moez.Cherif@inrialpes.fr` or `cherif@cs.sfu.ca`.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN (France)
Téléphone : 04 76 61 52 00 - International: +33 4 76 61 52 00
Télécopie : 04 76 61 52 52 - International: +33 4 76 61 52 52

Un algorithme de planification de mouvement pour la manutention d'objets dans des sites industriels évolutifs

Résumé : Ce papier traite du problème de la planification de mouvements sans collision pour une flotte de robots mobiles se déplacant dans un environnement évolutif. Nous nous concentrons sur le problème de planification de tâches de manutention d'objets : *trouver une trajectoire sans collision pour un groupe de robots mobiles pour bouger/déposer un ensemble d'objets amovibles dans un site industriel.* Notre algorithme traite de l'évolutivité de l'environnement en décomposant la planification en 2 phases : (1) la construction d'un graphe (dit "roadmap") qui capture la connectivité de espaces des configurations des différents systèmes mobiles, et (2) la planification de mouvements coordonnés sans collision pour ces systèmes et la manipulation adéquate et rapide de la roadmap pour traiter les changements de la structure de l'environnmement. L'algorithme a été implanté et testé sur des modèles de tâches de manutention 2D.

Mots-clé : planification de mouvement, manutention d'objets, systèmes multi-robots, robots mobiles, évitement de collision, environnements évolutifs.

1 Introduction

1.1 Overview of the paper

Robot motion planning has been extensively addressed during the last fifteen years yielding a large body of contributions (see [9] for a survey). These contributions focused on several instances of the general motion planning problem and concerned various types of tasks that can be achieved by a single robot (e.g., manipulation by an articulated arm, transportation by a mobile vehicle, dextrous manipulation by an artificial hand), or by multiple robots (e.g., manipulation by multi-arm systems, coordination of a fleet of mobile robots).

In this paper, we investigate an instance of the general motion planning problem for multiple robots within the application context of preparing and achieving tasks of handling materials in an industrial plant. The specific task we concentrate on is: *given a set of objects, find a collision-free coordinated path for a fleet of mobile (autonomously driven or human-operated) vehicles*[1] *for moving these objects from a starting position to a desired final position in the plant.* The nature of the handling operations to be carried out introduces new issues that have to be considered and makes this problem different from the classical coordinated motion planning problem for multiple robots. In the handling application context, and particularly when there are less robots than objects to be moved, handling operations need to be achieved sequentially. Each operation consists of moving an object by a robot, placing it at a desired position, and moving the robot toward an other object to be placed in the workspace. The sequential nature of handling the different objects has its effect in changing the structure of the workspace very often. This important feature has to be dealt with when designing a motion planner applicable to solve the handling tasks. The basic idea of our approach consists of pre-processing the C-spaces of all the moving systems (i.e., robots, and robot-object systems) and build a single representation – *a roadmap* – that captures the connectivity of the free regions of these C-spaces. Furthermore, the manipulation of this roadmap has to be easy and fast for accounting for the changes of the workspace. In addition, such a roadmap is used to find collision-free paths for the individual systems and to coordinate their motions safely. We have used a roadmap approach because it enables to plan repetitive handling operations taking place at the same workspace (as in our case). Unlike classical roadmap methods (such as a visibility graph method [12], or Voronoï graph-like roadmap [14], or the skeleton method [6]), we construct incrementally a roadmap that does not require any a priorly characterization of the free regions of the C-spaces of the moving systems. Considering recent results in randomized motion planning, we combine a random technique and a local discrete optimization method for searching the considered C-spaces and expanding a roadmap composed of a non-dense distribution of potential collision-free configurations (called *landmarks*) connected by simple safe paths. A particular care was taken to enable a shared use of the components of the roadmap by the different systems. The planner has been implemented and demonstrated on 2D models of industrial plants.

[1] For simplification, they are referred to by the term *robots*.

1.2 Other related works

An emerging approach for motion planning in static environments has been recently proposed by Kavraki and Latombe, and Svetska and Overmars [8, 15]. The approach, called *probabilistic path planning (PPP)*, consists of building randomly a dense roadmap that can be later used extensively for point-to-point fast path planning. It has been applied successfully to a variety of robot tasks as manipulation by a robot arm having a high number of degrees of freedom, path planning for non-holonomic car-like robot [15, 16], and coordination of the motion of multiple (car-like) robots [17]. Another probabilistic approach for path planning in high dimensional C-spaces is the *Ariadne Clew algorithm (ACA)* [1, 4]. Starting from the initial configuration, the algorithm expands a tree of landmarks connected by collision-free paths. At each time, a landmark that is the farthest to the tree is placed in the C-space and linked to the tree using a collision-free path finder based on genetic algorithms. Random techniques such as the *PPP* and the *ACA* have been shown to be suitable for robots with many degrees of freedom and C-spaces containing narrow passages between the C-obstacles. When the structure of the environment varies, a revision of these algorithms is needed for accounting for the changes of the robot C-space, hence of the roadmap. Recently, this problem has been addressed by Horsch *et al.* [7] and McLean *et al.* [11, 10]. In [7], a similar planner than *PPP* is used for a robot arm and the authors describe an algorithm for repairing the roadmap by reflecting the paths at the C-obstacles (an instance of the rebound technique [1, 8]). In [11, 10], the authors made use of the *ACA* to build a roughly dense roadmap for path planning for a robot arm. For coping with the workspace changing and its effect on the C-space, a mapping between these two spaces is built and used efficiently for detecting components of the roadmap that are in collision with new detected obstacles. The repairing process is based on a random generation of new paths and landmarks in the neighborhood of the new C-obstacle surfaces.

Path planning methods for a multiple-robot system fall into two main approaches: *centralized* and *decoupled* methods. Centralized methods consist of posing the motion planning problem in a composite C-space (i.e., the composition of the C-spaces of the different robots), and considering the multi-robot system as a single multi-body robot with a high number of degrees of freedom. Planning a collision-free path for the system can then be solved by applying any single-robot path planner (for instance the randomized potential-field planner of Barraquand *et al.* [3, 2]). The centralized approach has, in general, the advantage to be complete, but has a restricted applicability when the composite C-space is high dimensional mainly when a large number of robots have to be moved. Decoupled methods (as the priority-based method [5] and the path coordination technique [13]) are more practical, but not complete when solving complex problem such as planning collision-free paths in a workspace with many narrow passages. A more recent and emerging approach proposed in [17] consists of applying the probabilistic-roadmap paradigm to coordinate the motion of several moving objects. This approach is very close to the ideas we develop in this paper.

1.3 Organization of the paper

§2 states the addressed motion planning problem. §3, §4 and §5 present the roadmap construction and how it is used for planning handling tasks by a single and multiple robots, respectively. §6 describes simulation results.

2 The problem formulation

2.1 The task description

The considered handling tasks are assumed to take place in a workspace $\mathcal{W}$ (an industrial plant) described by a 2D area cluttered by walls with $n_{\mathcal{E}}$ entrance/exit doors $\mathcal{E}_i$ and containing at its interior $n_{\mathcal{B}}$ static polygonal obstacles $\mathcal{B}_i$ (see Fig. 1). The robot fleet is composed of $n_{\mathcal{A}}$ holonomic vehicles $\mathcal{A}_i$ that can freely translate and rotate. Each vehicle can move one of $n_{\mathcal{M}}$ objects, denoted by $\mathcal{M}_i$. The $\mathcal{A}_i$'s and the $\mathcal{M}_i$'s are all rigid and described by convex polygons.

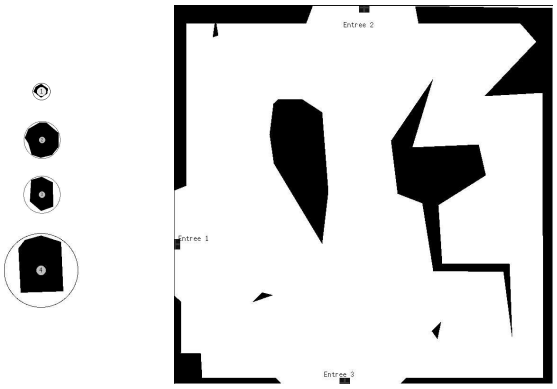

Figure 1: Left: 4 moving bodies ($\mathcal{A}_i$ and $\mathcal{H}_{i,j}$) and their approximating disks (3 disks are used), right: a workspace.

For a simple presentation of our approach, we consider the following assumptions: *(1) each $\mathcal{M}_j$ can be moved by any $\mathcal{A}_i$ under the condition that there exists at least a contact point between them and that they are rigidly connected through this contact (we denote by $\mathcal{H}_{i,j}$ the rigid system resulting from connecting $\mathcal{A}_i$ to $\mathcal{M}_j$)*, and *(2) each $\mathcal{A}_i$ and each $\mathcal{H}_{i,j}$ has a shape that it can be reasonably approximated by a disk $\mathcal{D}_k$ – the smallest disk it can be inscribed in (as shown in Fig. 1).* Assumption (2) allows to replace the $\mathcal{A}_i$'s and $\mathcal{H}_{i,j}$'s by their approximating disks and to restrict the range of their motion to translation only. This contributes in simplifying later the search of C-space and collision detection. Our subsequent research

aims to extend the approach described in this paper to account for the rotation motion of the different bodies. In addition and *wlog*, we consider that the $\mathcal{D}_k$'s (and hence the $\mathcal{A}_i$'s and the $\mathcal{H}_{i,j}$'s) are numbered in an increasing order so that $\mathcal{D}_1$ is the largest disk and $\mathcal{D}_{n_\mathcal{D}}$ is the smallest one, where $n_\mathcal{D}$ is the total number of disks used for the approximation.

2.2 The problem

Following the assumption (2), we consider that the configurations $q_{\mathcal{A}_i}$ of the $\mathcal{A}_i$'s and $q_{\mathcal{M}_i}$ of the $\mathcal{M}_i$'s are given by the positions in $\mathcal{W}$ of each body. $Q_\mathcal{A}$ and $Q_\mathcal{M}$ denote the configuration vector of the fleet of $\mathcal{A}_i$'s and $\mathcal{M}_i$'s, respectively. The handling problem is formally stated as:

> *Given a starting configuration $Q_{s,\mathcal{A}}$ for the $\mathcal{A}_i$'s and a starting configuration $Q_{s,\mathcal{M}}$ for the $\mathcal{M}_i$'s, all of them corresponding to collision-free positions located outside $\mathcal{W}$, and given a desired final configuration $Q_{g,\mathcal{M}}$ for the $\mathcal{M}_i$'s, find a collision-free trajectory for the $\mathcal{A}_i$'s enabling them to move the $\mathcal{M}_i$'s from $Q_{s,\mathcal{M}}$ to $Q_{g,\mathcal{M}}$ in a finite period of time.*

A solution to this problem has to provide for each object $\mathcal{M}_j$ which robot $\mathcal{A}_i$ will be in charge to move it, from each entrance $\mathcal{E}_k$ it will start moving it, and along which path it will take it toward its final goal $q_{g,\mathcal{M}_j}$. For focusing on the stated problem, we will not account for the motions of the different objects outside $\mathcal{W}$ assuming they move safely.

3 Roadmap generation

As we have pointed out in §1, the first stage of the planning algorithm aims to build a roadmap $\mathcal{G}$ that captures the connectivity of the C-spaces of the different $\mathcal{A}_i$'s and $\mathcal{H}_{i,j}$'s in a single representation. Due to the disk approximation applied to the moving systems, a connectivity graph built in the C-space for a given disk $\mathcal{D}_i$ can be also used to generate collision-free motions for disks (hence moving systems) equal or smaller than $\mathcal{D}_i$. We exploit the intersection between the different C-spaces to build a roadmap that can be used and shared locally by different systems.

3.1 Incremental tree building

We start building a connectivity representation of the C-space of the largest disk $\mathcal{D}_1$ (the largest moving system). The construction process is a revisited scheme of the landmark distribution used in the *ACA* (known as the *Explore* function [1, 4]), and operates as follows. At each door $\mathcal{E}_i$, we generate a landmark (i.e., a collision-free configuration) for $\mathcal{D}_1$ and

store it in a list LL_1. Knowing these landmarks, we search for a new landmark that is the farthest from them; i.e., the farthest to the closest one. This is achieved by applying a discrete maximization algorithm as follows. Let q_s be one of the landmarks initially in LL_1. Starting from $q_s = q(0)$ and a small increment of time δt, we compute among all possible motions of period δt the collision-free end-point configuration $q(\delta t)$ that maximizes the distance to overall the landmarks contained in LL_1. The set of possible motions is computed using a discrete sampling of motion control (direction and velocity) of $\mathcal{D}_1$ (see Fig. 2). This maximization scheme is iteratively repeated until $\mathcal{D}_1$ is deemed in a local maximum at a configuration $q(n\delta t)$, where n is the number of performed motion steps. This means that there are at least two landmarks of LL_1 which are at a minimum distance of $q(n\delta t)$, and no direction exists in which $\mathcal{D}_1$ can move while maximizing that distance. In this case, $q(n\delta t)$ is inserted in LL_1 and considered as a new landmark, and we sort LL_1 in a descending order considering the previous cost function. The path moving $\mathcal{D}_1$ to $q(n\delta t)$ is also stored. As only translation motions are considered, the path is represented by the set of configurations (positions) at which the direction of motion has been changed and the considered motion direction at that position. The first element of LL_1 is afterwards selected to generate a new landmark in the C-space of $\mathcal{D}_1$. When several landmarks have the same maximum cost in LL_1, we choose randomly one of them to be considered as a starting configuration. This process is iterated until a desired landmark density is achieved. Notice that because the randomness in selecting such landmarks, the obtained roadmap structure is not unique.

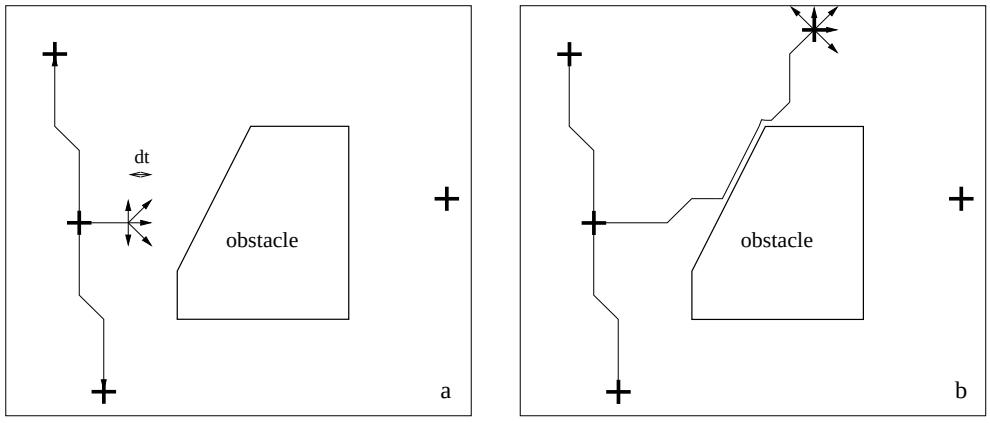

Figure 2: The two steps for generating a safe path

Depending on the structure of $\mathcal{W}$, the exploration of the C-space of $\mathcal{D}_1$ results in a set of $n_{\mathcal{E}}$ or less trees $\Upsilon_{1,j}$ where the nodes are the landmarks of LL_1 and the arcs are the paths between these landmarks. The roots of the $\Upsilon_{1,j}$'s are landmarks located on the doors $\mathcal{E}_k$ of $\mathcal{W}$. The nodes of $\Upsilon_{1,j}$'s are afterwards expanded in a similar way considering the disk $\mathcal{D}_2$ as the moving body. This expansion results in a set of new landmarks and paths located

in some forbidden regions of the C-space of $\mathcal{D}_1$ that are now reachable by $\mathcal{D}_2$. The same expansion process is iterated for all the disks; hence for all the $\mathcal{A}_i$'s and the $\mathcal{H}_{i,j}$'s. In the following, $\Upsilon_{i,j}$ denotes the trees of landmarks generated for $\mathcal{D}_i$. When $i \geq 2$, these have their roots at landmarks located on $\Upsilon_{i-1,j}$. Thus, a disk $\mathcal{D}_i$ can be moved safely along arcs of all the $\Upsilon_{k,j}$'s, for $k \leq i$. In Fig. 5, we show the different $\Upsilon_{i,j}$'s built for the 3 disks and the workspace shown in Fig. 1.

3.2 Transforming the trees into a roadmap

The second step of the algorithm consists of transforming the set of trees $\Upsilon_{i,j}$ into a more connected graph $\mathcal{G}$ (i.e., the roadmap). This step aims to connect landmarks located within a given neighborhood by generating new collision free paths moving the $\mathcal{A}_i$'s and the $\mathcal{H}_{i,j}$'s between them. For this purpose, we use a simple local planner based on a discrete minimization process operating in as similar way as the previously described local planner used for placing the landmarks (cf. §3.1). Given two selected landmarks, the local process starts from the initial landmark and minimizes, at each instant, the distance to the final landmark. Collision avoidance is locally achieved by "sliding" the moved system around the encountered obstacles (i.e., moving toward the left or the right of the obstacle).

In order to guarantee a reasonable number of edges in the roadmap, we only apply the local planner to connect couples of neighboring landmarks for which there is no existing path connecting them through the available graph and couples of landmarks for which there exists a connecting path and the length of this path is higher than a given measure. This measure incorporates a desired edge density and is written in the form: $k_{racc}d(l_1, l_2)$, where k_{racc} is a positive real number greater than 1 and d is the distance between two given landmarks l_1 and l_2. Figure 3 illustrates two roadmaps of different densities for the same set of vertices.

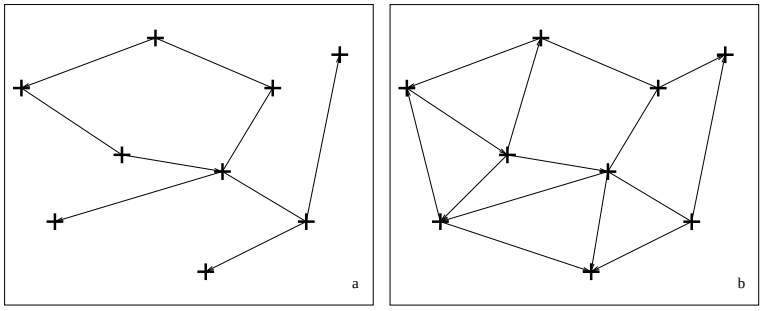

Figure 3: Left: low edges density. Right: higher edge density.

We have observed that the expansion of the trees used to build the roadmap results in winding paths connecting adjacent landmarks. For such paths, we apply an additional step which aims to minimize their length. It consists of a "smoothing" process which minimizes

the number of vertices used to describe a path (i.e., a configuration at which the direction of motion has been changed) while maintaining some topological features such as remaining in the same passage of the free space. Figure 4 illustrates such a smoothing step.

The resulting roadmap $\mathcal{G}$ is labeled by storing for each landmark and for each path the information concerning the largest moving system (or disk) that can be placed at that landmark or moved along that path while avoiding the obstacles $\mathcal{B}_i$. An illustration is given in Fig. 6.

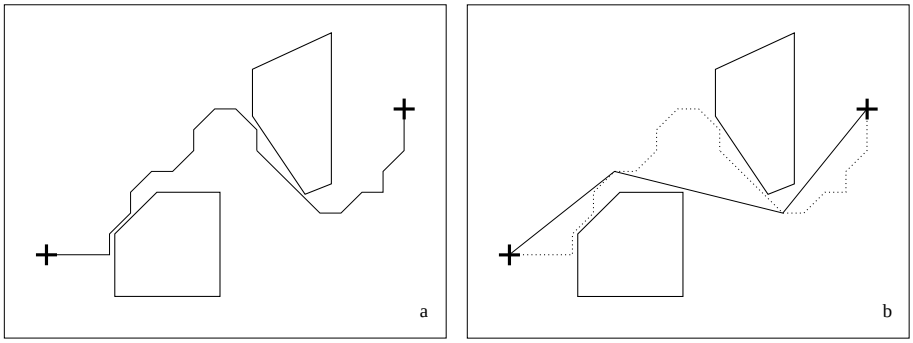

Figure 4: Left: initial path. Right: resulting smoothed path.

4 Path planning for a single robot

For developing our planning algorithm and for a simple presentation, we consider in this section the case where a single robot is used to move the different $\mathcal{M}_j$'s through $\mathcal{W}$. Let $\mathcal{A}_1$ be this robot.

4.1 Handling a single object

4.1.1 Path planning

Planning a collision-free path for moving an object $\mathcal{M}_j$ by a robot $\mathcal{A}_1$ between two configurations q_s and q_g is achieved by first by searching a path that moves the system $\mathcal{H}_{1,j}$ from q_s to a configuration q_1, and a path that takes it from q_g to q_2, where both q_1 and q_2 are lying on the same connected component of $\mathcal{G}$. These paths are planned by applying the local planner presented in §3. Once these paths are found, a path that connects q_1 to q_2 is searched for in $\mathcal{G}$. If it exists, the final path takes $\mathcal{H}_{i,j}$ from q_s to q_1, from q_1 to q_2 through $\mathcal{G}$, and from q_2 to q_g.

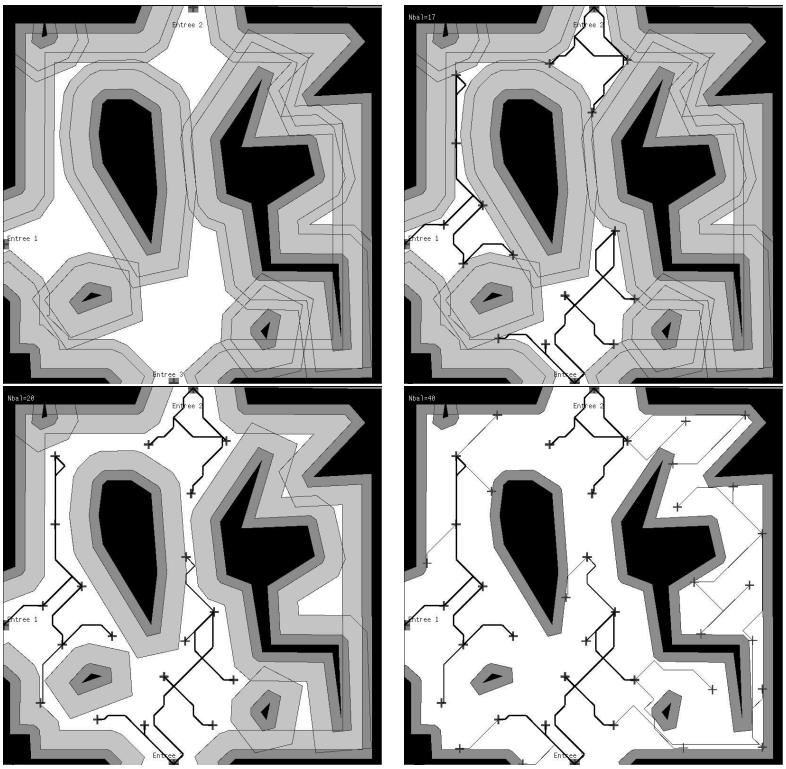

Figure 5: The free C-spaces (computed for illustration) of 3 different approximated systems (top left), and the expansion of the corresponding landmark trees (top right: $\mathcal{D}_3$, bottom left: $\mathcal{D}_2$, bottom right: $\mathcal{D}_1$).

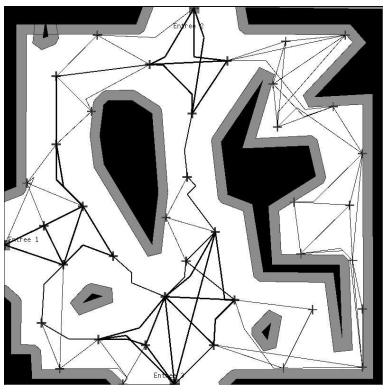

Figure 6: Building a graph from the trees.

4.1.2 Roadmap updating

After having placed the object $\mathcal{M}_j$ in $\mathcal{W}$, it is considered as a new obstacle. In this case, some components (i.e., landmarks and paths) of $\mathcal{G}$ become in collision with $\mathcal{M}_j$ leading in some situations to break the connectivity of $\mathcal{G}$. The updating process aims to repair the broken components of $\mathcal{G}$ by generating in it new collision-free landmarks and paths surrounding $\mathcal{M}_j$. The process of finding such new landmarks and paths is based on the same local process used for expanding the trees and transforming them into $\mathcal{G}$ (see §3). Fig. 7 illustrates the path planning step and the roadmap resulting after placing an object in $\mathcal{W}$.

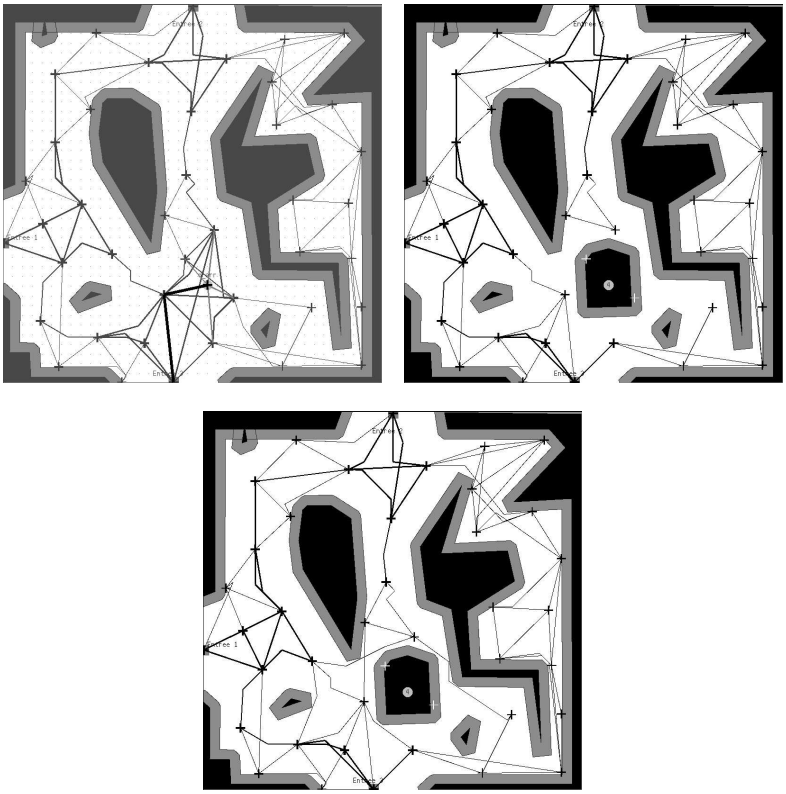

Figure 7: Top left: the planned path for $\mathcal{H}_{1,1}$ (dark lines). Top right: the resulting roadmap after adding $\mathcal{M}_1$. Bottom: the updated roadmap.

4.1.3 Scenario for moving a single object

The following scheme summarizes the solution of finding a safe path for moving a single object by a robot.

1. Use $\mathcal{G}$ to find a path that moves the robot-object system between a door and the desired location.
2. Place the object.
3. Exit the robot by moving it backward along the same path.
4. Update $\mathcal{G}$ considering the object as a new obstacle.
   (a) Eliminate landmarks/paths intersecting with the object (in C-space).
   (b) Generate new safe paths between landmarks surrounding the obstacle.

4.2 Handling all the objects

An intuitive solution consists of moving the objects sequentially to their desired configurations in $\mathcal{W}$ as follows. Using the roadmap $\mathcal{G}$, we plan a collision-free path for the system $\mathcal{H}_{1,1}$ between one of the entrances $\mathcal{E}_j$ to the configuration $q_{g,\mathcal{M}_1}$ of $\mathcal{M}_1$. Once $\mathcal{M}_1$ has been placed, we move $\mathcal{A}_1$ along the same path, but in the opposite direction until it reaches $\mathcal{E}_j$, and we update the roadmap $\mathcal{G}$ considering $\mathcal{M}_1$ as a new obstacle (cf. 4.1.2). Once this is achieved, $\mathcal{A}_1$ takes $\mathcal{M}_2$, and we try to find a collision-free path for the new system $\mathcal{H}_{1,2}$ that ends at the desired goal configuration $q_{g,\mathcal{M}_2}$ of $\mathcal{M}_2$. If this step is successfully achieved, we update parts of $\mathcal{G}$ in collision with $\mathcal{M}_2$, and we pursue iterating the same process for the remaining $\mathcal{M}_j, j = 3 \ldots n_{\mathcal{M}}$. When it fails, further trials are processed with different orders for moving the $\mathcal{M}_j$'s. This planning scheme can be time consuming because of the combinatoric to be considered which clearly increases with the number of entrances and objects to be moved in $\mathcal{W}$, and on the structure of the free C-space. Besides, it considers that once an object is placed in $\mathcal{W}$, it cannot be moved anymore. In order to overcome these drawbacks, we have developed a more elaborated planning scheme that makes use of the information embedded in $\mathcal{G}$ to find collision-free paths.

4.2.1 Planning using an influence graph

We solve the problem of handling several objects in the plant by a single robot as follows. For each system $\mathcal{H}_{1,j}$, we use $\mathcal{G}$ to find $n_{\mathcal{E}}$ collision-free paths $\pi_{j,k}$ ending at the final desired configuration of $q_{g,\mathcal{M}_j}$ and starting at a landmark located on the entrances $\mathcal{E}_k$. Each of these paths is searched while maximizing the following cost function:

$$f(\pi_{j,k}) = \min_{k=1\ldots n_{\mathcal{M}}, k\neq j} (d(\pi_{j,k}, q_{g,\mathcal{M}_k}))$$

where $d()$ is the minimum Euclidean distance in the C-space between a path and a configuration. Using such a cost function means that $\mathcal{H}_{1,j}$ moves along $\pi_{j,k}$ while being the farthest from the final goals of the other objects $\mathcal{M}_k$. Depending on the structure of $\mathcal{W}$ and $\mathcal{G}$, some

of the paths $\pi_{j,k}$ may be empty, i.e., there exist some entrances from which there is no path ending at a given $q_{g,\mathcal{M}_j}$. For a given object $\mathcal{M}_j$, the distance computation done for evaluating the cost function f allows us implicitly to determine which objects $\mathcal{M}_k$ intersect $\pi_{j,k}$ at their goals. In the case of an intersection, one has to move $\mathcal{M}_j$ before $\mathcal{M}_k$. Considering this observation, we build a direct graph $\mathcal{G}_{inf}$ where the nodes corresponds to the generated paths $\pi_{j,k}$ for all the $\mathcal{H}_{1,j}$'s, and the arcs encode the *influence* between these paths (see top of Figure 8). An arc points from a node $N(\pi_1)$ to a node $N(\pi_2)$ if the end-point of π_1 is in collision with at least a configuration of π_2. When the graph $\mathcal{G}_{inf}$ has no cycle, we start moving sequentially the objects corresponding to the sink nodes (i.e., nodes which are sources of no arc in $\mathcal{G}_{inf}$ and are pointed to by at least one arc) along the computed paths. If there are several sinks, we choose an arbitrary order (for instance randomly). Afterwards, we remove these sinks and the arcs pointing to them, and we characterize the set of new sinks in $\mathcal{G}_{inf}$. This scheme is iterated until $\mathcal{G}_{inf}$ becomes empty. Each time a system $\mathcal{H}_{1,j}$ has been moved and an object $\mathcal{M}_j$ has been placed at its goal, we update $\mathcal{G}$ for accounting for new forbidden regions in the free space due to the presence of $\mathcal{M}_j$. If $\mathcal{G}_{inf}$ has components composed of a single node, the corresponding objects are moved at the end.

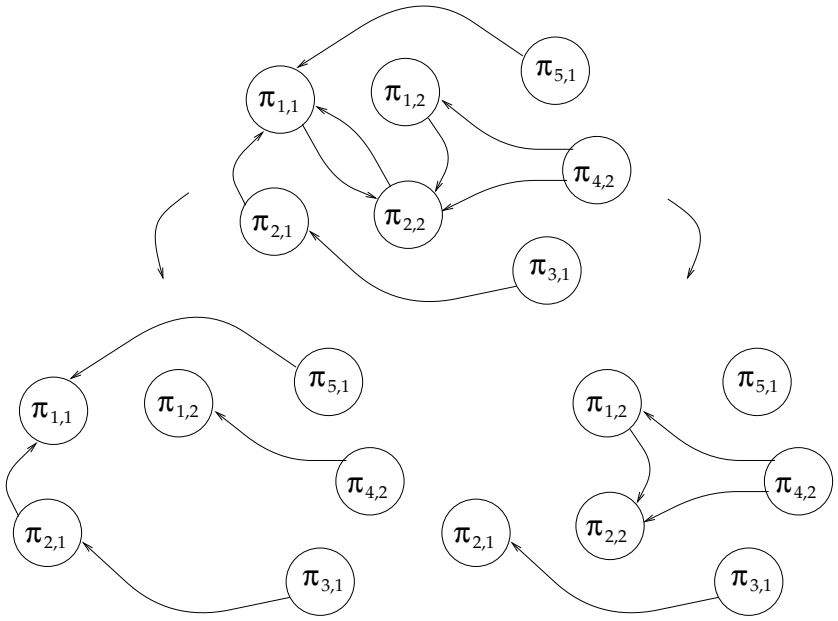

Figure 8: Top: An influence graph $\mathcal{G}_{inf}$. Bottom: two possible graphs resulting for removing the cycles of $\mathcal{G}_{inf}$.

4.2.2 Dealing with conflicts

When $\mathcal{G}_{inf}$ has cycles, we discard them by removing some nodes in $\mathcal{G}_{inf}$ and the corresponding arcs. This is achieved while maintaining at least one node (i.e., one path) for each object. If all the cycles has been removed successfully, we apply the same scheme as above to move sequentially the different objects. If a cycle remains, this means that a conflict will happen when handling the objects, and restricting a solution by simply coordinating the execution of the obtained paths is not sufficient. Assume that we have a conflict between two objects. It is solved by choosing one of them to be placed first by using the computed path as above. After having placed it and updated the roadmap $\mathcal{G}$, we search $\mathcal{G}$ for a new collision-free path for the second object. This path, if it exists, is different from the one provided by the scheme we described above. When such a path cannot be found, we reiterate the same scheme by inverting the order of placing the objects. If there is still no solution, the algorithm fails in finding a placement motion for one of these objects. As an illustration, consider the influence graph $\mathcal{G}_{inf}$ depicted in the top of Fig. 8 and computed for 5 systems $\mathcal{H}_{i,j}$ and 2 entrances. Discarding the cycles from $\mathcal{G}_{inf}$ leads to 2 possible new graphs shown at the bottom of Fig. 8. For instance, if the graph at the left bottom is considered, a possible solution to the handling problem consists of moving sequentially the different objects in the following order: $\mathcal{H}_{1,1}$ along $\pi_{1,2}$ (i.e., starting from entrance $\mathcal{E}_2$), $\mathcal{H}_{1,2}$ along $\pi_{2,1}$ (i.e., starting from entrance $\mathcal{E}_1$), $\mathcal{H}_{1,3}$ along $\pi_{3,1}$, $\mathcal{H}_{1,4}$ along $\pi_{4,2}$, and $\mathcal{H}_{1,5}$ along $\pi_{5,1}$.

4.2.3 Scenario for handling several objects by a single robot

The following scheme summarizes the solution of finding the paths and the order for moving several objects by a single robot.

1. For each robot-object system, use $\mathcal{G}$ to find a set of safe paths $\pi_{j,k}$ so that each $\pi_{j,k}$ is the farthest to the final locations of the other objects.
2. Build an *influence graph* $\mathcal{G}_{inf}$ as follows:
   - Node $N(\pi_{j,k})$ of $\mathcal{G}_{inf}$: safe path $\pi_{j,k}$ for the robot-object-j system between the door $\mathcal{E}_k$ and the desired location of object j.
   - Arc from $N(\pi_{j,k})$ to $N(\pi_{j',k'})$: the final location of $P_{j,k}$ intersects with at least one configuration of $P_{j',k'}$.
3. Remove the cycles from the influence graph $\mathcal{G}_{inf}$ and find an ordering for the handling operations:
   (a) If the resulting $\mathcal{G}_{inf}$ has no edges, any order for moving the objects along the $\pi_{j,k}$'s.

(b) If the resulting $\mathcal{G}_{inf}$ has edges (and possibly cycles), apply the following iterative scheme:

i. Choose a path π from a sink (or a random node of a cycle) of $\mathcal{G}_{inf}$.

ii. Move/place the object along π, update $\mathcal{G}$, and exit the robot along π.

iii. Remove the nodes/arcs of $\mathcal{G}_{inf}$ corresponding to the placed object.

5 Path planning for all the robots

In this section, we generalize the previous scheme for a handling task to be performed by all the fleet of $\mathcal{A}_i$. Let us consider the previously depicted influence graph $\mathcal{G}_{inf}$ and let us consider that the 5 objects has to be moved by 3 robots of the same size and shape. When, the paths do not use the same component of $\mathcal{G}$ other than possibly the final configuration of one of the involved objects, $\mathcal{G}_{inf}$ can be used to coordinate the motions of the different systems. For instance, according to the reduced graph at the bottom left, $(\mathcal{H}_{1,1}, \mathcal{H}_{2,2}, \mathcal{H}_{3,3})$ can be moved first and in parallel along the paths $(\pi_{1,2}, \pi_{2,1}, \pi_{3,1})$, and then $(\mathcal{H}_{1,4}, \mathcal{H}_{2,5})$ along $(\pi_{4,2}, \pi_{5,1})$. Assume now that we solve the same task using 5 robots. A possible motion is to start moving the different bodies in the same order than before, but we have to account for conflict situations; i.e., when the planned paths for the moving systems have some intersecting configurations or segments of path.

5.1 Processing the conflicts

A conflict between a robot and a placed object is processed as follows. For instance, if $\mathcal{M}_4$ is placed at its final goal $q_{g,\mathcal{M}_4}$ before $\mathcal{A}_1$ crosses this configuration when moving back along $\pi_{1,2}$, $\mathcal{A}_1$ is assumed to be able to grasp $\mathcal{M}_4$, to move around while staying at the same position $q_{g,\mathcal{M}_4}$, to put down $\mathcal{M}_4$ at that position, and to pursue its motion along $\pi_{1,2}$. In the current implementation, this assumption is considered only when there is no path exiting $\mathcal{A}_1$ through the updated roadmap. Remember that when an object is placed, $\mathcal{G}$ is updated for repairing broken components. This restricts the conflicts to 2 types: a conflict between 2 robots $\mathcal{A}_i$ and $\mathcal{A}_j$, and a conflict between a robot $\mathcal{A}_i$ and a handling system $\mathcal{H}_{j,k}$. The case where 2 robots $\mathcal{A}_i$ and $\mathcal{A}_j$ are in conflict is solved as follows. Let π_i and π_j be the planned paths along which $\mathcal{A}_i$ and $\mathcal{A}_j$ have to move, respectively. By construction, π_i and π_j have necessary their end-point at one of the exit doors. A simple coordination scheme we use consists of choosing (randomly) one of the 2 robots to move first along the common segment of π_i and π_j (or the segment corresponding to the colliding regions of $\mathcal{W}$ swept by the 2 robots). Let $\mathcal{A}_i$ be the chosen robot. If $\mathcal{A}_j$ moves later on the common segment and on the remaining parts of π_j, further conflicts can happen between the 2 robots if the two paths have other common segments. For avoiding this, we move $\mathcal{A}_j$ also along π_i (i.e., the same as $\mathcal{A}_i$). When the conflict concerns a robot $\mathcal{A}_i$ and a system $\mathcal{H}_{j,k}$, the idea is to exchange the handled object $\mathcal{M}_k$ between the robots and moving it by $\mathcal{A}_i$. Assume that the conflict

happens in a corridor, and let π_i and π_j the planned paths for $\mathcal{A}_i$ and $\mathcal{H}_{j,k}$, respectively. In order to avoid searching for a new path for one of the moving systems avoiding the conflict, $\mathcal{A}_j$ will put down $\mathcal{M}_k$ and move back to its exit along π_j (in opposite direction). $\mathcal{A}_i$ can then grasp $\mathcal{M}_k$ and move it towards its final goal.

5.2 Path generation

Assuming that the displacements along the edges of $\mathcal{G}$ last a period of time equal to 1, the parameterization of the execution of a given path π_i by a robot is achieved by defining a map m_p that numbers the landmarks and the arcs of $\mathcal{G}$ crossed by the robot. Such a numbering allows to encode the sequentiality of the execution of the components of different paths. The previous above-described motion schemes are embedded in a graph search aiming to build the maps m_p for each robot. In a similar way as the idea of *super-graph* used by Svetska and Overmars [17], we build incrementally a graph $\mathcal{SG}$ where a node corresponds to a placement where each moving system (robot or system of type $\mathcal{H}_{i,j}$) is located at a collision-free landmark of $\mathcal{G}$. The starting node of $\mathcal{SG}$ corresponds to a collision-free placement determined from the influence graph $\mathcal{G}_{inf}$. An arc of $\mathcal{SG}$ exists if at least one system can move from one landmark to another safely considering the motion schemes we developed in the previous sections and accounting for the changes of $\mathcal{W}$ (hence the updates performed on $\mathcal{G}$). The cost of each arc is 1. During the transition between the different nodes, we account also for the fact that a given system $\mathcal{H}_{i,j}$ can be transformed into a simple robot $\mathcal{A}_i$ (after placing the object $\mathcal{M}_j$) and vise-versa. Starting from the initial node, $\mathcal{SG}$ is expanded by the search algorithm until a final desired placement (i.e., a node in $\mathcal{SG}$) is reached. In such a placement, each robot is located on a landmark on one of the exit doors and all the objects are placed at their goals in $\mathcal{W}$. When it is found, a solution is a collision-free path of minimal length (i.e., crossing a minimal number of graph edges) allowing the robots to perform safely the desired handling task. The maps m_p can then be built while extracting the final solution (i.e., the paths of the n robots) from $\mathcal{SG}$. When at a given iteration, no new nodes can be generated in $\mathcal{SG}$, this means that the search space has been fully explored and that a final desired placement cannot be reached. In this case, the algorithm fails in finding a solution to the handling problem.

5.2.1 Scenario for handling the object by multiple robots

As described above, the solution to the handling problem by multiple robots consists of solving a sequence of multiple-system problems, i.e., handling by using simultaneously $1 \leq n \leq n_{\mathcal{A}}$ robots, where n is the number of sinks (and/or random nodes of cycles) in $\mathcal{G}_{inf}$. The following scheme summarizes the construction and the search of the direct graph $\mathcal{SG}$.

Starting from the initial placement given by the initial point of the n paths $\pi_{i,j}$ chosen from $\mathcal{G}_{inf}$, *expand* a direct graph, $\mathcal{SG}$, where:

- each node is a collision-free placement of the n systems on the nodes of $\mathcal{G}$,
- each arc corresponds to k, $1 \leq k \leq n$, collision-free instantaneous motions on the edges of $\mathcal{G}$ (there is no landmark shared by 2 systems, and there are no systems exchanging their current landmarks),

until a final node is reached; i.e., a placement where the n robots are on the exit doors and the objects are in their desired positions in $\mathcal{W}$.

The following treatments are considered during each iteration of the search:

1. Node expansion: generate new nodes (placements) corresponding to simultaneously moving the n systems along 1 edge of the $\pi_{i,j}$'s (each system can move or stay at rest).
2. Placing an object: after having placed object j,
   (a) update $\mathcal{G}$ and move back robot j along $\pi_{i,j}$, and
   (b) update the intersecting $\pi_{i',j'}$'s (or plan new paths).
3. Local treatment of the conflict situations during the expansion of $\mathcal{SG}$:
   (a) **Conflict between the robot i and the object j'** (this case happens when $\mathcal{G}_{inf}$ has a cycle). When there is no valid updated path for the robot i, it is assumed that it can re-grasp the object and turn around it before exiting.
   (b) **Conflict between two robots i and i'** (this case happens when both the robots are going to exit). One robot is prioritized randomly to exit first and the second one is moved along the same path.
   (c) **Conflict between the robot-object system $i-j$ and the robot i'**. The object is exchanged between the two robots. This consists of forming a new system $i'-j$. Once the exchange is achieved, a path is planned to exit the robot i.
   (d) Other possible cases of conflicts are solved by the combinatorics of the search.

6 Implementation and results

The planning algorithm has been implemented with C++, and simulation examples have been carried out on a Silicon Graphics Indy R4400 Workstation. These concerned handling objects in an industrial environment by a fleet of mobile holonomic robots. We present examples illustrating the applicability of our approach for coping with conflicts and planning handling operations performed by a single and multiple robots.

In Fig. 9, we consider the task of moving 2 robots (approximated by the same disk) in a workspace $\mathcal{W}$ reduced to a corridor. The top left snapshot shows the built roadmap $\mathcal{G}$ using 4 landmarks and 5 collision-free paths. In the top right and middle left of the figure, this roadmap has been used to move sequentially the robot $\mathcal{A}_1$ and $\mathcal{A}_2$ towards their goals. When the object moved by $\mathcal{A}_2$ is placed, two path segments of $\mathcal{G}$ become forbidden. Since, there is no components in $\mathcal{G}$ enabling $\mathcal{A}_1$ to exit from $\mathcal{W}$, a new arc is generated (by the roadmap updating step) (see the middle right snapshot) and used for exiting $\mathcal{A}_1$ as shown in the bottom snapshot.

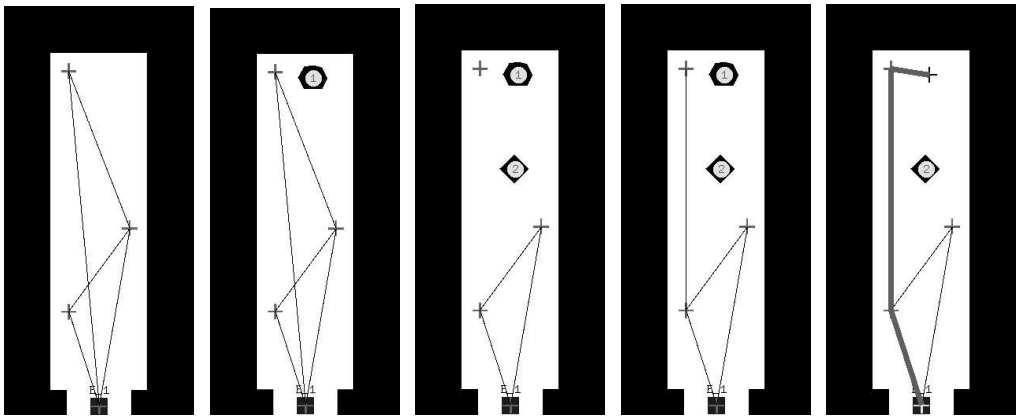

Figure 9: A planned path allowing a robot to avoid a placed object in a corridor.

The example in Figure 10 concerns the case of placing 5 objects $\mathcal{M}_j$ (shown at the left) by a single robot. The generated roadmap is shown on the middle and have been computed considering that the 5 objects are approximated by 3 different disks. The final configurations of the objects are shown at the right. We considered in this example that the robot has the same size and shape as the smaller object and that when it handles one of the $\mathcal{M}_j$, it is completely included in it. The use of the influence graph $\mathcal{G}_{inf}$ yields to perform the following sequence of treatment (shown in Figures 11 and 12 from left to right, and from top to bottom): *(1) move* $\mathcal{H}_{1,1}$ *along* $\pi_{1,2}$ *(starting from the right entrance* $\mathcal{E}_2$*), place* $\mathcal{M}_1$*, and move back* $\mathcal{A}_1$ *along* $\pi_{1,2}$ *towards* $\mathcal{E}_2$*, (2) update* $\mathcal{G}$*, (3) move* $\mathcal{H}_{1,2}$ *along* $\pi_{2,1}$*, place* $\mathcal{M}_2$*, and exit* $\mathcal{A}_1$ *from the left entrance* $\mathcal{E}_1$*, (4) update* $\mathcal{G}$*, (5) move* $\mathcal{H}_{1,3}$ *along* $\pi_{3,1}$*, place* $\mathcal{M}_3$*, and exit* $\mathcal{A}_1$ *from* $\mathcal{E}_1$*, (6) update* $\mathcal{G}$*, (7) move* $\mathcal{H}_{1,4}$ *along* $\pi_{4,2}$*, place* $\mathcal{M}_4$*, and exit* $\mathcal{A}_1$ *from* $\mathcal{E}_2$*, (8) update* $\mathcal{G}$*, (9) move* $\mathcal{H}_{1,5}$ *along* $\pi_{5,1}$*, place* $\mathcal{M}_5$*, and exit* $\mathcal{A}_1$ *from* $\mathcal{E}_1$*, and (10) update* $\mathcal{G}$. The resulting updated roadmap is shown at the bottom right. When the $\mathcal{M}_j$'s have to be moved by 5 robots (assumed to be of the same size as the smaller object), the obtained solution consists of moving first $(\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3)$ along their corresponding paths (with $\mathcal{M}_3$ starting after $\mathcal{M}_2$ since they use the same segments in $\mathcal{G}$), and second $(\mathcal{M}_4, \mathcal{M}_5)$. Note that for the robot handling $\mathcal{M}_2$, the algorithm needed to plan a new exiting path allowing to avoid $\mathcal{M}_3$ when this is placed at its final position. Note that for performing this solution, only 3 robots can be located at the same time in $\mathcal{W}$. In Fig. 13, we show results obtained by

processing the same task considering a slightly denser roadmap (i.e., having a larger number of landmarks and paths segments by decreasing the increment δt (cf. §3)). The obtained coordinated solution allows to move the 5 objects in the same time in $\mathcal{W}$ while avoiding collision. Unlike the first solution, all the 5 robots can move in $\mathcal{W}$ at the same time and exit safely avoiding conflict situations.

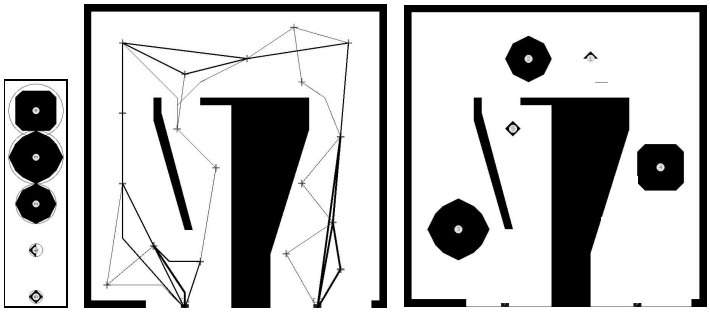

Figure 10: Left: 5 objects and their approximation by disks. Middle: Workspace and roadmap for placing the 5 objects. Right: Final desired positions of the objects.

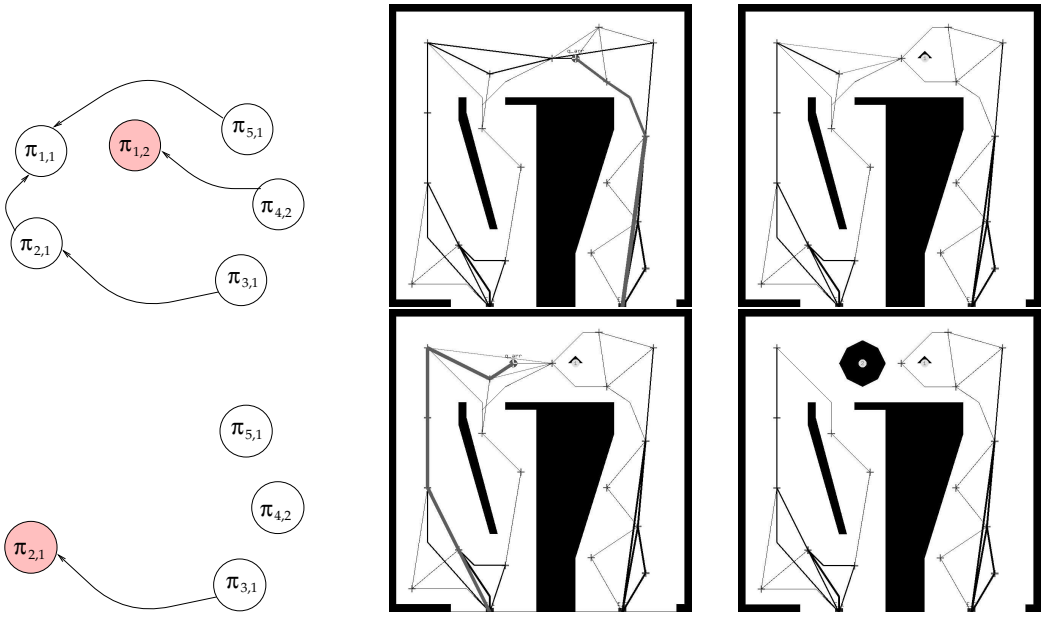

Figure 11: Sequential placement of objects showing the ability of the algorithm to avoid conflict situations by using the influence graph $\mathcal{G}_{inf}$.

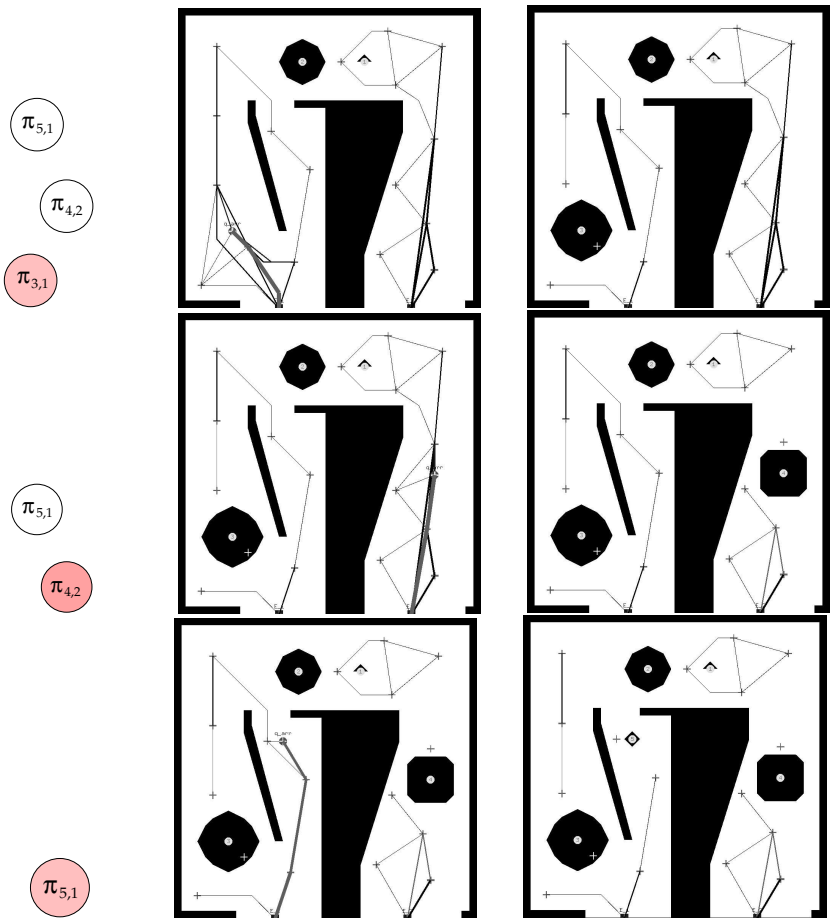

Figure 12: Sequential placement of objects (continued).

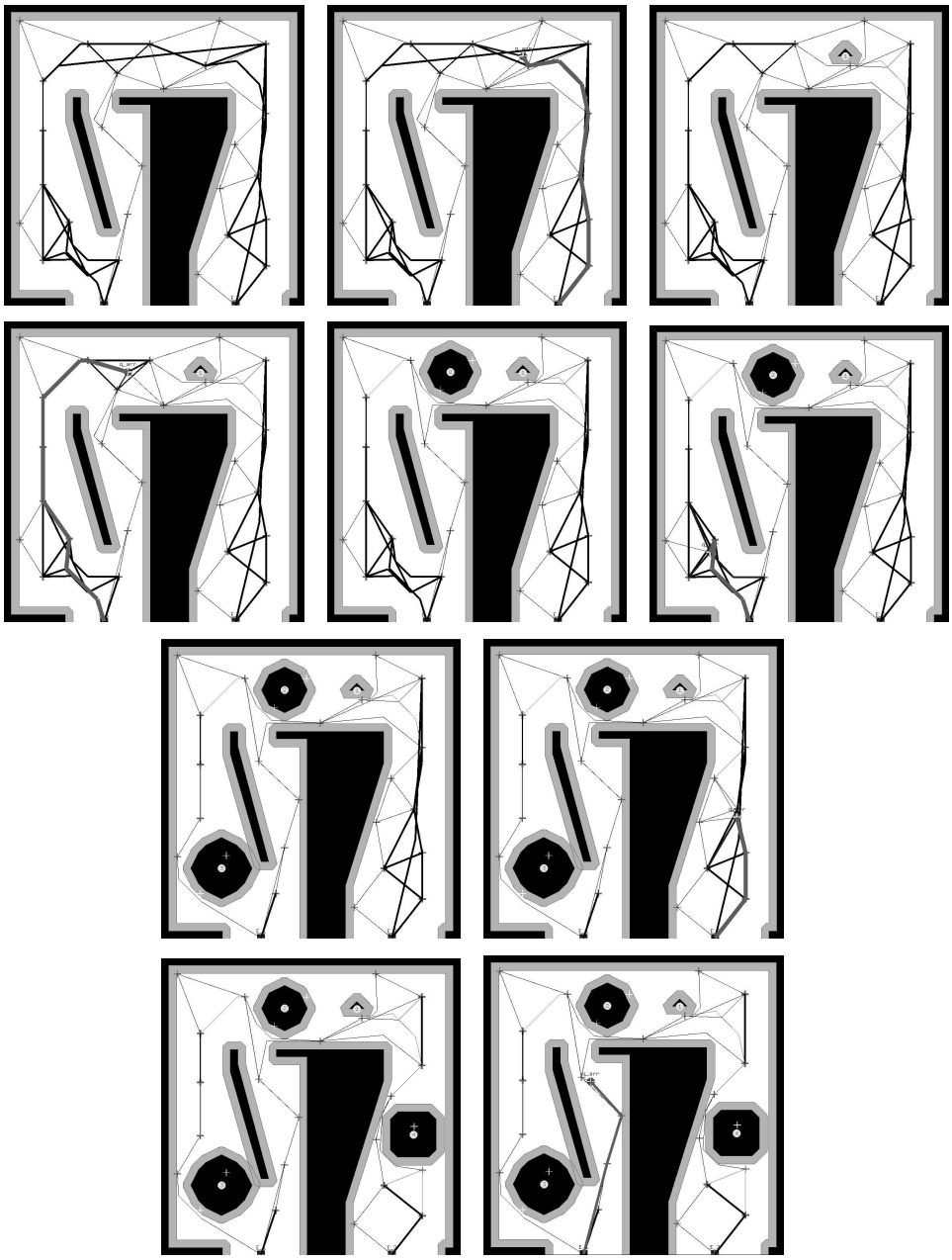

Figure 13: Planned paths allowing the coordination motion of several objects.

7 Conclusion

In this paper, we addressed developing a high-level tool based on motion planning applicable to the preparation of handling/intervention tasks (i.e., moving a set of object by a fleet of mobile robots) in a changing industrial plant. Our approach is based on a roadmap approach consisting of (1) pre-processing the C-spaces of the moving systems for providing a single graph −the roadmap− that captures the connectivity of these C-spaces, and (2) using this roadmap for planning collision-free paths for the fleet of handled objects when accounting for the changes of the workspace. The main features of our planning algorithm is that it makes use of a rapid scheme to update the roadmap (when objects are placed in the workspace) and it makes use of a novel idea −the influence graph− to cope with conflicts in both the case of handling the objects by a single robot and multiple robots. The different schemes we have described led to a first implementation that has demonstrated the applicability of our approach for different handling cases. For the multiple robots case, we are currently investigating how to use in a more practical way the swept surfaces by the different paths when coordinating the motions of several moving systems and enhancing the applicability of the planner to solve much more complex handling tasks. Other extensions to be addressed concern the incorporation of the orientation of the moving systems and more general shapes of the objects.

References

[1] J.-M. Ahuactzin-Larios. *Le Fil d'Ariane : Une Méthode de Planification Génerale. Application à la Planification Automatique de Trajectoires.* Thèse de doctorat, Inst. Nat. Polytechnique de Grenoble, Grenoble (F), Sep. 1994.

[2] J. Barraquand, B. Langlois, and J-C Latombe. Numerical potential field techniques for robot path planning. Research Report STAN-CS-89-1285, Robotics Lab, Computer Science Dept, Stanford Univ., CA (USA), Oct. 1989. 38 pages.

[3] J. Barraquand and J-C. Latombe. Robot motion planning: a distributed representation approach. Research Report STAN-CS-89-1257, Robotics Lab, Computer Science Dept, Stanford Univ., CA (USA), May 1989.

[4] P. Bessière *et al.* Planning in a continuous space with forbidden regions : The ariadne's clew algorithm. In K. Goldberg, D. Halperin, J-C. Latombe, and R. Wilson, editors, *Algorithmic Fondations of Robotics.* A.K. Peters Publisher, Wellesley, MA (USA), 1995.

[5] S.J. Buckley. Fast motion planning for multiple moving robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, pages 322–326, Scottsdale, AZ (USA), May 1989.

[6] J. F. Canny and M. C. Lin. An opportunistic global motion planner. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, pages 1554–1559, Cincinnatti, OH (USA), May 1990.

[7] Th. Horsch, F. Schwarz, and H. Tolle. Motion planning with many degrees of freedom − random reflections at c-spaces obstacles. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, pages 3318–3323, San Diego, CA (USA), May 1994.

[8] L. E. Kavraki, P. Svetska, J-C Latombe, and M. H. Overmars. Probalistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robotics and Automation*, 12(4):566–580, Aug. 1996.

[9] J-C. Latombe. *Robot motion planning.* Kluwer Academic Press, 1991.

[10] A. McLean and Ch. Laugier. Update and repair of a roadmap after model error discovery. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, volume 2, pages 917–924, 1996.

[11] A. McLean and I. Mazon. Incremental roadmaps and global path planning in evolving industrial environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, volume 1, pages 101–107, Minneapolis, (USA), Apr. 1996.

[12] N.J. Nilsson. A mobile automaton: an application of artificial intelligence techniques. In *Proc. of the Int. Joint Conf. on Artificial Intelligence*, pages 509–520, Washington, DC (USA), 1969.

[13] P. A. O'donnell and T. Lozano-Péres. Deadlock-free and collision-free coordination of two robot manipulators. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, pages 484–489, Scottsdale, AZ (USA), May 1989.

[14] C. Ó'Dúnlaing and C. Yap. A retraction method for planning the motion of a disc. *Journal of Algorithms*, 6:104–111, 1982.

[15] M. H. Overmars and P. Svetska. A probabilistic learning approach to motion planning. In K. Goldberg, D. Halperin, J-C. Latombe, and R. Wilson, editors, *Algorithmic Fondations of Robotics.* A.K. Peters Publisher, Wellesley, MA (USA), 1995.

[16] A. Scheuer and Th. Fraichard. Continuous-curvature path planning for car-like vehicles. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 997–1003, 1997.

[17] P. Svetska and M. H. Overmars. Coordinated motion planning for multiple car-like robots using probabilisitc roadmaps. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, Nagoya (J), May 1995.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399